R. I. Damper (Ed.) Data-Driven Techniques in Speech Synthesis. Kluwer, 149-172, 2001

TreeTalk: Memory-based word phonemisation

Walter Daelemans Antal van den Bosch*

Abstract

We propose a memory-based (similarity-based) approach to learning the mapping of words into phonetic representations for use in speech synthesis systems. The main advantage of memory-based data mining techniques is their high accuracy, the main disadvantage is processing speed. We introduce a hybrid between memory-based and decision-tree-based learning (TRIBL) which optimises the trade-off between efficiency and accuracy. TRIBL was used in TREETALK, a methodology for fast engineering of word-to-phonetics conversion systems. We also show that for English, a single TRIBL classifier trained on predicting phonetic transcription and word stress at the same time performs better than a 'modular' approach in which different classifiers corresponding to linguistically relevant representations such as morphological and syllable structure are separately trained and integrated.

1 Introduction

A central component in a typical text-to-speech system is a mapping between the orthography of words and their associated pronunciation transcription in some phonetic alphabet with diacritics (denoting, e.g., syllable boundaries and word stress). We call this mapping *phonemisation*. Table 1 provides examples in a few languages of input and output of a phonemisation module.

In a traditional knowledge engineering approach, several linguistic processes and knowledge sources are presupposed to be essential in achieving this task. The MITALK system (Allen, Hunnicutt & Klatt, 1987) is a good example of this approach. The architecture of their phonemisation module includes explicitly implemented, linguistically-motivated abstractions such as morphological analysis, rules for spelling changes at morphological boundaries, phonetic rules, etc. Each of these modules and their interaction have to be handcrafted, and redone for each new language, language variant, or type of phonetic transcription. Also, lists of exceptions, not covered by the developed rules have to be collected and listed explicitly.

The obvious problems with such an approach include high knowledge engineering costs for development of a single phonemisation module, limited robustness and adaptability of modules

* ILK, Computational Linguistics, Tilburg University

Table 1: Sample words with their phonetic transcription (including stress markers). Samples are from Dutch, British English, French, and Scottish Gaelic, respectively.

word (input)	*phonetic transcription (output)*
belofte	/bə'lɔftə/
biochemistry	/"baɪəʊ'kɛmɪstrɪ/
hésiter	/ezi'te/
cabar	/'kapər/

once they are developed, and lack of reusability for additional languages or dialects. Data-oriented (data mining) techniques using statistical or machine learning techniques have therefore recently become increasingly popular (starting with Stanfill and Waltz, 1986 and Sejnowski and Rosenberg, 1987; see Damper, 1995 for a recent overview of self-learning approaches). Advantages of these systems include fast development times when training material is available, high accuracy, robustness, and applicability to all languages for which data in the form of machine readable pronunciation dictionaries are available.

In previous research (Van den Bosch and Daelemans, 1993; Daelemans and Van den Bosch, 1993; Daelemans and Van den Bosch, 1996; Van den Bosch, 1997), we applied memory-based techniques (Stanfill and Waltz, 1986; Aha *et al.*, 1991) with good results to the phonemisation task in British and American English, Dutch, and French. Memory-based learning is also referred to as **lazy learning** (see Aha, 1997 for an overview of theory and applications of lazy learning techniques and applications) because all training examples are stored in memory without any further invested effort in processing at learning time. No abstractions are built at learning time from the examples as is the case in **eager learning** techniques such as rule induction, top down induction of decision trees, connectionism, and statistical methods. At processing time, a new input is compared to all examples in memory, and the output is extrapolated from those memory items that are most similar to the input according to some similarity function. For the phonemisation task, when using a well-designed similarity function, memory-based techniques turned out to have a higher generalization accuracy (i.e. accuracy on words the system was not trained on) than a number of eager approaches (Van den Bosch, 1997), a result which generalizes to a large number of natural language processing tasks. Empirically, it can be shown that in learning natural language processing tasks, abstraction is harmful to generalization accuracy (see Daelemans, 1996; Van den Bosch, 1997 for tentative explanations). Furthermore, resulting trained systems are highly competitive when compared to knowledge-based handcrafted systems (Van den Bosch and Daelemans, 1993).

Unfortunately, memory-based techniques are computationally expensive, as (i) all examples have to be stored in memory, and (ii) every input pattern has to be compared to all memory items in order to find the most similar items. Existing indexing and compression techniques either only work well for numeric feature values, or they involve some form of abstraction (e.g. when using induced decision trees as an index to the case base). We cannot use the former because the

phonemisation task will require symbolic (nominal) feature values only, and we have established empirically that the latter are harmful to generalization accuracy. This article addresses this issue and proposes a solution in which an optimal tradeoff can be found between memory and processing efficiency on the one hand, and processing accuracy on the other hand.

In Section 2, we relate our own previous memory-based phonemisation approach to other research on using analogical reasoning for phonemisation, and provide empirical results comparing the approach to various other learning approaches. In Section 3, we introduce the TRIBL learning algorithm which allows dynamic selection of a tradeoff between efficiency and accuracy. Subsection 3.2 reports on empirical results for phonemisation in English using TRIBL in the TREETALK system. We conclude with a discussion (based on Van den Bosch, 1997; Van den Bosch *et al.*, 1997) of the interaction of linguistic modules that is optimal in a memory-based learning approach to phonemisation.

2 Memory-Based Phonemisation

To determine the phonemisation of a word given its spelling, it is worthwhile looking at the words with a similar spelling of which the phonemisation is known. Words with similar spellings have similar phonemisations. The 'new' word veat is pronounced /vit/ because it resembles among others the known words heat, feat, eat, veal, and seat in all of which the substrings (v)ea(t) are pronounced /(v)i(t)/. This type of memory-based reasoning has been acknowledged by many researchers as a useful approach in building phonemisation modules. It pays, however, to keep in memory the phonemisations of all known words, in order to prevent that a word like great be pronounced /grit/ instead of /greɪt/. Together, these insights suggest an approach in which (i) a lexicon of words with their associated phonemisations is kept in memory, and (ii) a similarity-based method is used to compute the phonemisations of words not in the lexicon.

MBRTalk (Stanfill and Waltz, 1986; Stanfill, 1987) was the first application of this approach to phonemisation. Their **memory-based reasoning** is a paradigm which places recall from memory at the foundation of intelligence. Every letter in every word is represented as a record in the database. Each record contains a fixed number of fields (features), in this case features are the spelling symbol to be transcribed, the transcription, the four letters to the left and to the right (a window of 9), and a stress marker. A novel test word is entered as a record with some fields filled (the predictors; the context in this case) and some fields empty (the goals: transcription and stress marker). This target is compared to each frame in the database, using a computational measure of (dis)similarity, and the most similar records are used to extrapolate the value for the goal field. The approach is therefore an adaptation for symbolic (non-numeric) features of the k-NN algorithm (Cover and Hart, 1967; Hart, 1968; Gates, 1972; Dasarathy, 1980) from statistical pattern recognition.

The most basic distance metric for patterns with symbolic features is the **overlap metric** given in Equations 1 and 2; where $\Delta(X, Y)$ is the distance between patterns X and Y, represented by

n features, w_i is a weight for feature i, and δ is the distance per feature.

$$\Delta(X,Y) = \sum_{i=1}^{n} w_i \, \delta(x_i, y_i) \tag{1}$$

where:

$$\delta(x_i, y_i) = 0 \; if \; x_i = y_i, \; else \; 1 \tag{2}$$

This metric, which we will call IB1 in this paper, is obviously deficient when different features have a different relevance for solving the task, as is the case in phonemisation. The metric Stanfill and Waltz, 1986 propose is therefore more sophisticated: it combines a feature relevance metric with a value difference metric.

Feature relevance metrics assign weights to features according to their relevance in solving the task. For example, in phonemisation, the target grapheme to be transcribed is obviously the most relevant feature, and should therefore be assigned more weight than the context features. Value difference metrics assign different distances or similarities to different pairs of values of the same feature. Again in phonemisation, graphemes f and v are more similar than f and a, and when matching the feature values of a new record with those of the database, this should be taken into account. The specific value difference metric used by Stanfill and Waltz was simplified and made symmetric later by Cost and Salzberg, 1993.

Stanfill and Waltz reported superior generalization accuracy to NETtalk (Sejnowski and Rosenberg, 1987), the famous neural network backprop learning approach to phonemisation, on the same data. In Weijters (1991), NETtalk was compared to a more simple memory-based reasoning procedure using a weighted overlap metric. Feature relevance weights were hand-set, descending in strength from the target grapheme to be described towards context features further away. NETtalk was also outperformed by this approach. In Wolpert (1990) a similar memory-based learning experiment was performed with the same results.

In Van den Bosch and Daelemans (1993), the overlap metric was combined with **information gain**, an information-theoretic feature relevance weighting method, inspired by Quinlan's (1986, 1993) work on top-down induction of decision trees. Information gain was also used by Lucassen and Mercer (1984) to guide a decision tree building process for learning phonemisation. Information gain (IG) weighting looks at each feature in isolation, and measures how much information it contributes to our knowledge of the correct class label. The information gain of feature f is measured by computing the difference in uncertainty (i.e. entropy) between the situations without and with knowledge of the value of that feature (Equation 3).

$$w_f = H(C) - \sum_{v \in V_f} P(v) \times H(C|v) \tag{3}$$

Where C is the set of class labels, V_f is the set of values for feature f, and $H(C) = -\sum_{c \in C} P(c) \log_2 P(c)$ is the entropy of the class labels. The probabilities are estimated from relative frequencies in the training set. The resulting IG values can then be used as weights in Equation 1. The k-NN algorithm with this metric is called IB1-IG (Daelemans and Van den Bosch, 1992).

While our approach focuses on classification of single graphemes in context to their phonemic mapping, other approaches have been proposed that map **variable-width chunks** of graphemes to chunks of phonemes by analogy (Sullivan and Damper, 1992; Sullivan and Damper, 1993; Pirelli and Federici, 1994; Yvon, 1996; Damper and Eastmond, 1997) most of which build forth on Glushko's (1979) psycholinguistically-oriented single-route model of reading aloud, and Dedina and Nusbaum's (1991) PRONOUNCE model for chunk-based text-to-speech conversion.

2.1 Memory-based phonemisation with IGTree

The most important problem with memory-based approaches is their time and space complexity. Storing and using all patterns extracted from all words in the lexicon with their associated phoneme is prohibitively inefficient unless implemented on massively parallel machines (Stanfill and Waltz, 1986). In Daelemans and Van den Bosch (1996), we presented an optimized approximation of memory-based learning using IGTREEs. In this section we will describe the design of this phonemisation approach, its general philosophy, and the results of empirical comparisons. Section 3 shows how IGTREE can be seen as one end of a continuum and 'normal' lazy learning (IB1-IG) as the other, and how an optimal efficiency–versus–accuracy tradeoff can be found between these extremes.

The approach we take for phonemisation could be described as **lexicon-based generalization**. What we need is a datastructure from which we can retrieve known phonemisations of words, and at the same time extrapolate to new, previously unseen words by their similarity to the words in the lexicon. Our solution is to base generalization on the capacity of our algorithm to automatically learn to find those parts of words on which similarity matching can safely be performed. The system stores single grapheme–phoneme correspondences with a minimal context that is sufficient to be certain that the mapping is unambiguous (in the lexicon given). These induced contexts range from very small (corresponding to general rules) to very large (corresponding to lexical structures), and everything in between. E.g. for English, a v followed by o is transcribed as a /v/ (a general rule), and an i preceded by es and followed by den is transcribed into /ɪ/, a very specific context necessary to make the distinction between the pronunciation of grapheme i in president and preside.

In Daelemans and Van den Bosch (1996), the following general architecture for the TREETALK word phonemisation system was proposed to implement this approach. Figure 1 illustrates the architecture visually.

Automatic alignment. The spelling and the phonetic trancription of a word often differ in

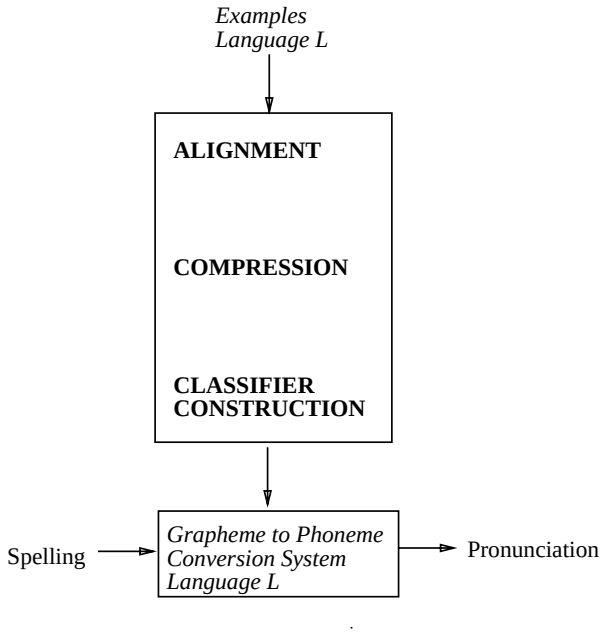


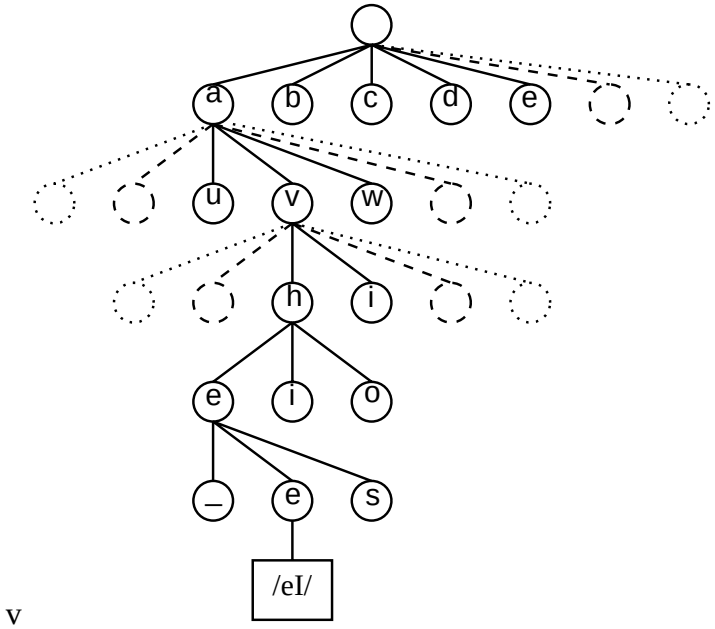
Figure 1: General architecture of the TREETALK word phonemisation system.

length (e.g. rookie - /ruki/). Our phonemisation approach demands, however, that the two representations be of equal length, so that each individual graphemic symbol can be mapped to a single phonetic symbol (e.g. r oo k ie - /r u k i/). Our algorithm tries to align the two representations by adding **null** phonemes in such a way that graphemes or strings of graphemes are consistently associated with the same phonetic symbols (e.g. rookie - /ru–ki–/, where the '–' depicts a phonemic null). Our algorithm works in a purely probabilistic way by capturing all possible phoneme-grapheme mappings with their frequency, and selecting the most probable mapping at word-pronunciation pair level on the basis of the multiplied individual letter-phoneme association probabilities. Empirical tests indicate that there is no significant loss in generalization accuracy when using our automatic alignment instead of a handcrafted one.

Alternative approaches to alignment are possible. For example, Van Coile (1990) uses a Hidden-Markov phoneme model in conjunction with the Viterbi algorithm (Viterbi, 1967); Luk and Damper (1996) combine stochastic grammar induction with Viterbi decoding using a maximum likelihood criterion.

Lexicon compression. In our approach, all lexical data (all available aligned spelling-pronunciation pairs) have to be kept in memory, but since (i) we only need for each letter in each word the minimal context necessary to find a unique corresponding phoneme, and (ii) the same grapheme-context-phoneme combinations may reappear in many words, a considerable amount of compression is possible. To achieve this compression, we use IGTREEs (Daelemans *et al.*, 1997a). An IGTREE is a decision tree in which the root node represents the feature with the

highest relevance for solving the task, the nodes connected to the root the feature with the second most highest relevance, etc. The arcs connecting a node to the nodes at the level below represent the different values attested for that feature in the lexicon. In the phonemisation application, paths in the IGTREE represent grapheme-context-phoneme combinations: connections between the root node and the next level represent the graphemes to be classified, connections between other levels of the tree represent context graphemes in order of relevance, and leaf nodes represent the phonemes associated with the grapheme-context paths. Figure 2 shows part of an IGTREE for English phonemisation.



v

Figure 2: Retrieval of the pronunciation of a, of the word behave. The path represents the minimally disambiguating context ehave.

To determine the relevance order of graphemes, we used information gain (see Equation 3), which we adopted from other decision tree learning methods (e.g., Quinlan, 1993). However, mutual information has been proposed in phonemisation as early as Lucassen and Mercer (1984). Not surprisingly, the information gain (and hence relevance) of the grapheme to be transcribed is highest, followed by the nearest context graphemes to the left and the right, with context letters further away becoming decreasingly important. The depth of a path reflects in a certain sense the ambiguity of the mapping it represents. Leaves near the top of the decision tree typically belong to highly regular pronunciations.

The result of this compression using IGTREE is that all transcriptions in the lexicon can be retrieved from this datastructure very quickly, and that the lexicon is compressed into a fraction of its original size.

Generalizing from the compressed lexicon. To retrieve the phonemisation of a word that was not in the learning material, each letter of the new word is taken as a starting point of tree search.

The search then traverses the tree, up to the point where the search successfully meets a leaf node, or where the search fails as the specific context of the new word was not encountered in the learning material, and consequently was not stored as a path in the tree. In the first case, the phonemic label of the leaf node is simply taken as the phonemic mapping of the new word's letter. In the second case, the exact matching strategy is taken over by a **best guess** strategy. In present implementations of the system, the best guess strategy is implemented in a straightforward way. When building a path in the tree, the construction algorithm constantly has to check whether an unambiguous phonemic mapping has been reached. At each node, the algorithm searches in the learning material for all phonemic mappings of the path at that point of extension. In cases when there is more than one possible phonemic mapping, the algorithm computes what is the most probable mapping at that point. Computation is based on occurrences: the most frequent mapping in the learning material is preferred (in case of ties, a random choice is made). This extra information is stored with each non-ending node. When a search fails, the system returns the most probable phonemic mapping stored in the node at which the search fails.

Finally, the phonemisation of a new word is assembled by concatenating IGTREE's phonetic classifications of each of its graphemes in context.

Decision trees for phonemisation have been introduced by Lucassen and Mercer (1984), who also used an information-theoretic metric as a guiding principle in constructing the tree. Their context features are preceding and following letters and preceding phonemes, coded as binary features, which necessitates a recursive search for the most informative binary features using mutual information. The application of Machine Learning algorithms such as ID3 and C4.5 (Quinlan, 1986; Quinlan, 1993) to phonemisation is analogous to this approach (see e.g. Dietterich *et al.*, 1995; Ling and Wang, 1996). IGTREE differs from these approaches in that it keeps all information in the training data, necessary for classification, in the tree, unlike the previous approaches which prune or restrict the tree in order to improve generalization. In IGTREE, generalization is achieved by the way the word level problem is cut up into letter level parts, and by the defaults computed on the nodes of the tree.

The IGTREE approach, although developed independently with a completely different motivation, also turns out to be equivalent to Kohonen's (1986) Dynamically Expanding Context approach (DEC), applied to phonemisation in Torkkola (1993). DEC extracts rules from the data according to a predefined specificity hierarchy (in this case, specificity of context and the intuition that context further away from the focus position is decreasingly relevant). Starting with rules transforming a single grapheme into a phoneme, context is added until, given a particular training set, the rules extracted are unambiguous in the training set, i.e. the context and grapheme combination determine a unique phoneme. As the rules are stored as a tree structure, the resulting method is almost identical to the Van den Bosch & Daelemans (1993) version of IGTREE. The only difference is that in IGTREE, information gain is used to automatically decide upon the 'specificity hierarchy' instead of handcrafting it. Finally, an approach to phonemisation similar to Van den Bosch and Daelemans (1993) and Torkkola (1993) has been used in the Onomastica project for the transcription of proper names (Andersen and Dalsgaard, 1994).

Table 2: Examples of instances generated for task GS from the word booking.

instance number	*left context*			*focus letter*	*right context*			*classification*
1	_	_	_	b	o	o	k	/b/1
2	_	_	b	o	o	k	i	/u/0
3	_	b	o	o	k	i	n	/-/0
4	b	o	o	k	i	n	g	/k/0
5	o	o	k	i	n	g	_	/ɪ/0
6	o	k	i	n	g	_	_	/ŋ/0
7	k	i	n	g	_	_	_	/-/0

2.2 Experiments: applying IGTree to word phonemisation

We base our experiments on a corpus of English word pronunciations, extracted from the CELEX lexical data bases (Van der Wouden, 1990; Burnage, 1990) maintained at the Centre for Lexical Research of the University of Nijmegen. The corpus contains 77,565 unique word pronunciations. Each listed word pronunciation is accompanied by additional information on morphological analysis, syllabification, and stress assignment. We define the phonetic mapping of a grapheme in context as its associated phoneme plus a stress marker indicating whether the phoneme is an initial phoneme of a syllable receiving primary stress (marker '1'), secondary stress (marker '2'), or no stress (marker '0'). This task definition thus integrates grapheme-phoneme conversion and stress assignment in one task; we refer to it henceforth as the GS task.

Table 2 displays example instances and their combined phoneme/stress marker classifications generated on the basis of the sample word booking. The phonemes with stress markers are denoted by composite labels. For example, the first instance in Table 2, ___book, maps to class label /b/1, denoting a /b/ which is the first phoneme of the syllable /bu/ receiving primary stress.

On the basis of the CELEX corpus of 77,565 words with their corresponding phonemic transcription with stress markers, a data base containing 675,745 instances was constructed. The number of actually occurring combinations of phonemes and stress markers is 159, which is fewer than the (Cartesian) product of the number of occurring subclasses ($62 \times 3 = 186$); some phoneme/stress marker combinations do not occur in the data.

To allow for a comparison between memory-based (lazy) learning algorithms and alternative learning algorithms, we selected the lazy-learning algorithms IB1 (Aha *et al.*, 1991; Daelemans *et al.*, 1997a) and IB1-IG (Daelemans and Van den Bosch, 1992; Daelemans *et al.*, 1997a), and IGTREE (Daelemans *et al.*, 1997a), the optimised decision-tree approximation of IB1-IG, all described above. Moreover, we selected two additional eager learning algorithms. First, we selected C5.0, an updated implementation of C4.5 (Quinlan, 1993). C4.5 is a decision tree algorithm which is similar to IGTREE, but which recomputes information gain at each added non-ending node, and which is supplied with various additional features. From these features,

results summary	
best algorithm	IB1-IG
gen. error (%)	6.28
gen. error, words (%)	37.88

overlap and bias	
overlap error (%)	20.10
class bias error (%)	81.16

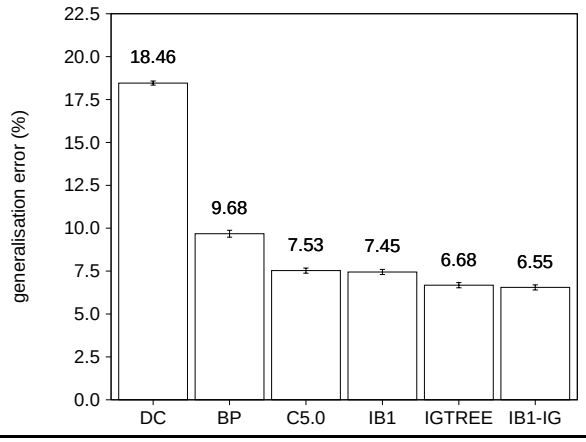


Figure 3: Generalization accuracy results of IB1, IB1-IG, IGTREE, C5.0, BP, and DC on the GS task. Results summary, bias and overlap errors (left), and generalization errors in terms of the percentage incorrectly classified test instances of five algorithms (right).

we selected subsetting of values, and default pruning (for details, cf. Quinlan, 1993). Second, we selected the connectionist learning algorithm Back-propagation learning (Rumelhart *et al.*, 1986), henceforth BP, which is a form of eager learning since it spends a considerable amount of effort in recoding the training material as real-valued weights of connections between simple processing units. BP is run on a three-layered perceptron, using local codings of input ($7 * 42 = 294$ units) and output (159 units), and 200 hidden units (for details, cf. Van den Bosch, 1997). Finally, to provide a baseline accuracy, the simple DC algorithm (Van den Bosch, 1997) was also selected for application to the GS task: DC classifies new instances either by copying the classification of duplicate instances in memory when found (i.e., exploiting the overlap between training material and new material), or, when the latter fails, by guessing the most frequent classification in the whole memory (i.e., exploiting the bias in the training material to the best uninformed guess).

Figure 3 displays all generalization accuracies in terms of incorrectly classified test instances, a results summary, and the overlap and bias errors. A test instance is classified incorrectly when the phoneme part is misclassified, or the stress-marker part, or both. Performance is measured in terms of generalization accuracy, i.e., the precision with which test instances are classified. In Figure 3, accuracies are displayed as generalization errors, i.e., percentages of misclassified test instances.

The results displayed in Figure 3 indicate that IB1-IG performs best on test instances. The generalization accuracy of IB1-IG is slightly but significantly better than that of IGTREE (a one-tailed t-test yields $t(19) = 1.94, p < 0.05$). All other differences are significant with $p < 0.001$, the smallest difference being between IGTREE and IB1 ($t(19) = 11.48$, $p < 0.001$). Thus, memory-based learning yields adequate performance when it is augmented with an information-gain weighted similarity metric, in IB1-IG. IGTREE, the approximate optimisation of IB1-IG, generates a comparable, yet significantly higher number of errors than IB1-IG. While eager

Table 3: Average memory storage (in kilobytes) and processing time per test instance (in seconds) obtained with IB1-IG and IGTREE, applied to the GS dataset.

algorithm	*storage (Kb)*	*time per instance (s)*
IB1-IG	4,751	0.4826
IGTREE	760	0.0052

learning in IGTREE is competitive with IB1-IG, eager learning in C5.0 and BP results in considerably lower accuracies than that of IB1-IG. Both algorithms abstract from their learning material more eagerly than IGTREE does, either by explicit forgetting of feature-value information through pruning (in C5.0) or by compressed encoding (in BP). Abstraction as performed by C5.0 and BP appears harmful for generalization performance for this task.

In sum, when optimal accuracy is desired for the GS task, the results point to IB1-IG as the most suited algorithm. However, when computational efficiency is also desired in the target application, there appears to be a trade-off between IB1-IG and IGTREE. Table 3 displays the memory storage (in terms of kilobytes needed for storing all feature-value and class information) and processing costs (in terms of seconds needed to classify one test instance, on average[1]) of both algorithms. The slight advantage in generalization accuracy of IB1-IG over IGTREE is at the cost of considerably more memory storage; IGTREE's memory compression compared to that of IB1-IG is 84.0%. Moreover, IGTREE is about 93 times faster than IB1-IG.

3 TRIBL and TREETALK

The results discussed in the previous section show that, for the phonemisation task, there is a slight (yet significant) performance loss when using IGTREE compared to pure memory-based learning as used in IB1-IG. For some other tasks, in which the relevance of the predictive features cannot be ordered in a straightforward way, IB1-IG or even IB1 may perform significantly better than IGTREE; e.g., when applied to morphological segmentation and stress assignment of English words (Van den Bosch *et al.*, 1996; Van den Bosch, 1997). Clearly, what is needed is an algorithm that allows us to find an optimal position (from the point of view of the user of the resulting system) between the extremes of pure memory-based learning with feature weighting (IB1-IG) which is most accurate but least efficient, and the IGTREE approach which is most efficient, but may in some case result in lower generalization accuracy.

[1] Processing times were measured on a PC equipped with a Pentium 75 Mhz processor running Linux.

3.1 The TRIBL learning method

The TRIBL method is a hybrid generalization of IGTREE and memory-based learning in IB1-IG. TRIBL searches for an optimal trade-off between (i) minimisation of storage requirements by building an IGTREE (and consequently also optimisation of search time) and (ii) maximal generalization accuracy. To achieve this, a parameter is set determining the switch from IGTREE to memory-based learning in IB1-IG, in learning as well as in classification. The parameter value denotes the last level at which IGTREE compresses homogeneous subsets of instances into leaves or non-terminal nodes; below this level, all remaining uninspected feature values of instances in non-homogeneous subsets (i.e., instances that are not yet disambiguated) are stored uncompressed in **instance-base nodes**. During search, the normal IGTREE search algorithm is used for the levels that have been constructed by IGTREE; when IGTREE search has not ended at the level marked as the switch point between IGTREE and IB1-IG, one of the instance-base nodes is accessed, and IBL-IG is employed on the sub-instance-base stored in this instance-base node.

3.2 Experiments with TREETALK

The architecture of the phonemisation system TREETALK described in Subsection 2.1 was modified to use TRIBL instead of IGTREE (trained on the same aligned data)[2]. The empirical results are shown in Table 4. Again, the results were achieved by means of 10-fold CV experiments on the full GS data. The table repeats the results obtained with the pure memory-based learning methods IB1 and IB1-IG, the results with IGTREE, and lists in between the results obtained with different values for the various TRIBL switch points from IGTREE to memory-based learning. We show both accuracy and efficiency in terms of storage needed and processing time.

The results listed in Table 4 show that TRIBL indeed offers a trade-off between memory storage costs and processing time on the one hand, and generalization accuracy on the other hand. As regards memory storage costs, the table shows a gradual decrease of the memory needed; the memory costs of IB1-IG can be reduced to about 50% (in the case of TRIBL-3) without significant loss of performance. With TRIBL-3, processing time is reduced considerably, and is hardly different from IGTREE's processing time. Even TRIBL-1 obtains a marked reduction in processing time, without any loss in performance; generating nodes representing the values of the most important feature (i.e., the middle grapheme) partitions the full instance base in 42 considerably smaller subsets (one for each letter of the alphabet[3]); with each search, only one out of 42 subsets needs to be searched to still find the best classifications.

While we have tested all possible switch points between IGTREE and IB1-IG, we have also investigated two heuristics for finding an optimal switch point automatically. The results in Table 4 suggest that TRIBL-3 maintains IB1-IG's accuracy as well as IGTREE's speed.

[2] See Daelemans *et al.*, 1997b for a discussion of results of TRIBL on several non-linguistic benchmark datasets.

[3] The alphabet in the English CELEX corpus contains 26 letters, twelve letters with diacritics, and the three non letter characters ., ', and -.

Table 4: Average generalization performance (with standard deviation, after the $\pm$ symbol), of IGTREE, IB1, IB1-IG, and TRIBL, with memory storage and processing time per test instance, applied to the GS dataset.

algorithm	*generalization accuracy (%)*	*storage (Kb)*	*time per instance (s)*
IB1	92.54±0.16	4,751	0.4386
IB1-IG	93.45±0.15	4,751	0.4826
TRIBL-1	93.45±0.15	4,157	0.0075
TRIBL-2	93.45±0.15	3,551	0.0054
TRIBL-3	93.43±0.15	2,683	0.0053
TRIBL-4	93.34±0.15	1,601	0.0052
TRIBL-5	93.32±0.15	912	0.0052
IGTREE	93.32±0.15	760	0.0052

First, we assert that the **branching factors** per level in IGTREE-constructed trees (i.e., the fan-out of feature-value tests per node per level) offer both an explanation for the performance of the various TRIBLs as well as a potentially adequate switch point heuristic. The higher the branching factor of a node at a certain level in the tree, the smaller the subsets represented by the node's child nodes, i.e., the less consumptive the search. Alternatively, when a node has a low branching factor, it may not pay off to create daughter nodes, as compared to leaving the subset uncompressed, to be searched with a memory-based search strategy. In the trees generated by IGTREE on the GS data, the first two levels have high average branching factors: 42 (i.e., the maximal branching factor) and 18, respectively. The lower five levels display average branching factors ranging from 4 to 1; the first two levels are clearly outlyers. TRIBL-3 optimally reflects this branching factor difference: it builds a two-level tree on the basis of the two most important features, and leaves the remaining five feature values uncompressed. Since TRIBL-3 indeed matches the accuracy of IB1-IG and the processing speed of IGTREE, it can be said to be an optimal TRIBL for the GS task.

Second, we have experimented with a heuristic function for estimating an optimal switch point between IGTREE and IB1-IG based on **average feature information gain**; when the information gain of a feature exceeds the sum of the average information gain of all features + one standard deviation of the average, then the feature is used for constructing a decision tree level (Daelemans *et al.*, 1997b). This function points to TRIBL-1 as optimal, since only the information gain of the middle grapheme is markedly higher than the threshold value, while all other features fall below this value. This switch point heuristic points at a safer switch point than the branching factor heuristic. More systematic experiments are needed to determine the efficacy of these two heuristics.

With IGTREE and TRIBL, processing efficiency is sufficiently high to allow experimenting with architectures containing several separately trained modules, micmicking the architecture of tra-

ditional knowledge-based systems which included modules for morphological analysis, syllabification etc. In the next section, we investigate the efficacy of such modularisation for obtaining optimal generalization accuracy.

4 Modularity and linguistic representations

Mainstream phonological and morphological theories, influenced by Chomskyan linguistic theory across the board since the publication of SPE (Chomsky and Halle, 1968), have generally adopted the idea of abstraction levels in various guises (e.g., levels, tapes, tiers, grids) (Goldsmith, 1976; Liberman and Prince, 1977; Koskenniemi, 1984; Mohanan, 1986) as essential abstractions in modelling morpho-phonological processing tasks, such as word phonemisation.

According to these leading morpho-phonological theories, systems that (learn to) convert spelled words to phonemic words in one pass, i.e., without making use of abstraction levels, such as the learning algorithms trained on the GS task, are assumed to be unable to generalize to new cases: going through the relevant abstraction levels is deemed essential to yield correct conversions. This assumption implies that if one wants to build a system that converts text to speech, one should implement explicitly all relevant levels of abstraction. As indicated earlier, such explicit implementations of abstraction levels can indeed be witnessed in many state-of-the-art speech synthesisers, implemented as (sequential) modules (Allen *et al.,* 1987; Daelemans, 1988).

This strong claim prompts the question whether the generalization accuracy of IB1-IG, IGTREE, or TRIBL on the GS task could be surpassed by systems which employ abstraction levels explicitly. We performed systematic experiments to investigate this question (for a full description of all experiments, cf. Van den Bosch, 1997). We trained and tested a word-phonemisation system reflecting linguistic assumptions on abstraction levels quite closely: the model is composed of five sequentially-coupled modules. Second, we trained and tested a model in which the number of modules was reduced to three, integrating two pairs of levels of abstraction. Third, we compared the generalization accuracies of both modular systems with the accuracy results obtained on the GS task, reported in Section 3.

The resource of word-phonemisation instances used for these experiments is again the CELEX lexical data base of English (Burnage, 1990). We added to our corpus of 77,565 words with their pronunciations with stress markers, all information related to these words' morphological segmentation, syllabification, and stress assignments. Each abstraction level is computed in each modular system by a module; each module performs a different morpho-phonological subtask. Analogous to our procedure for constructing instances for the GS task, for each subtask an instance base is constructed containing instances produced by windowing and attaching to each instance the classification appropriate for the (sub)task under investigation. Table 5 displays example instances derived from the sample word booking. For each (sub) task an instance base of 675,745 instances is built.

Table 5: Examples of instances generated from the word booking, with classifications for all of the subtasks investigated, viz. M, A, G, Y, S, and GS.

	letter-window instances											*phoneme-window instances*							
instance	*left*			*right*			*classifications*					*left*				*right*			*classif.*	
number	*context*		*focus*	*context*			M	A	G	S	GS	*context*		*focus*	*context*			Y	S
1	_	_	_	b	o	o	1	1	/b/	1	/b/1	_	_	_	/b/	/u/	/-/	/k/	1	1
2	_	_	b	o	o	k	0	1	/u/	0	/u/0	_	_	/b/	/u/	/-/	/k/	/ɪ/	0	0
3	_	b	o	o	k	i	0	0	/-/	0	/-/0	_	/b/	/u/	/-/	/k/	/ɪ/	/ŋ/	0	0
4	b	o	o	k	i	n	0	1	/k/	0	/k/0	/b/	/u/	/-/	/k/	/ɪ/	/ŋ/	/-/	1	0
5	o	o	k	i	n	g	1	1	/ɪ/	0	/ɪ/0	/u/	/-/	/k/	/ɪ/	/ŋ/	/-/	_	0	0
6	o	k	i	n	g	_	0	1	/ŋ/	0	/ŋ/0	/-/	/k/	/ɪ/	/ŋ/	/-/	_	_	0	0
7	k	i	n	g	_	_	0	0	/-/	0	/-/0	/k/	/ɪ/	/ŋ/	/-/	_	_	_	0	0

In the table, six classification fields are shown, one of which is a composite field; each field refers to one of the (sub)tasks investigated here. M stands for morphological decomposition; A is graphemic parsing[4]; G is grapheme-phoneme conversion; Y is syllabification; S is stress assignment, and GS is integrated grapheme-phoneme conversion and stress assignment. The example instances in Table 5 show that each (sub)task is phrased as a classification task on the basis of windows of letters or phonemes (the stress assignment task S is investigated with both letters and phonemes as input). As with the GS task, each window represents a snapshot of a part of a word or phonemic transcription, and is labelled by the classification associated with the middle letter of the window. For example, the first letter-window instance ___book is linked with label '1' for the morphological segmentation task (M), since the middle letter b is the first letter of the morpheme book; the other instance labelled with morphological-segmentation class '1' is the instance with i in the middle, since i is the first letter of the (inflectional) morpheme ing. Classifications may either be binary ('1' or '0') for the segmentation tasks (M, A, and Y), or have more values, such as 62 possible phonemes (G) or three stress markers (primary, secondary, or no stress, S), or a combination of these classes (159 combined phonemes and stress markers, GS).

4.1 Experiments

Our experiments are grouped in three series, each involving the application of IGTREE to a particular word-phonemisation system. IGTREE is selected rather than IB1-IG or TRIBL, since it is more efficient for the construction of a multi-module system; it is not feasible to hold five instance bases in memory at the same time, while keeping five decision trees in memory simultaneously

[4]Graphemic parsing is not represented in the CELEX data. We used the automatic alignment algorithm discussed earlier (cf. Daelemans and Van den Bosch, 1996) to determine which letters are the first or only letters of a grapheme (class '1') or not (class '0').

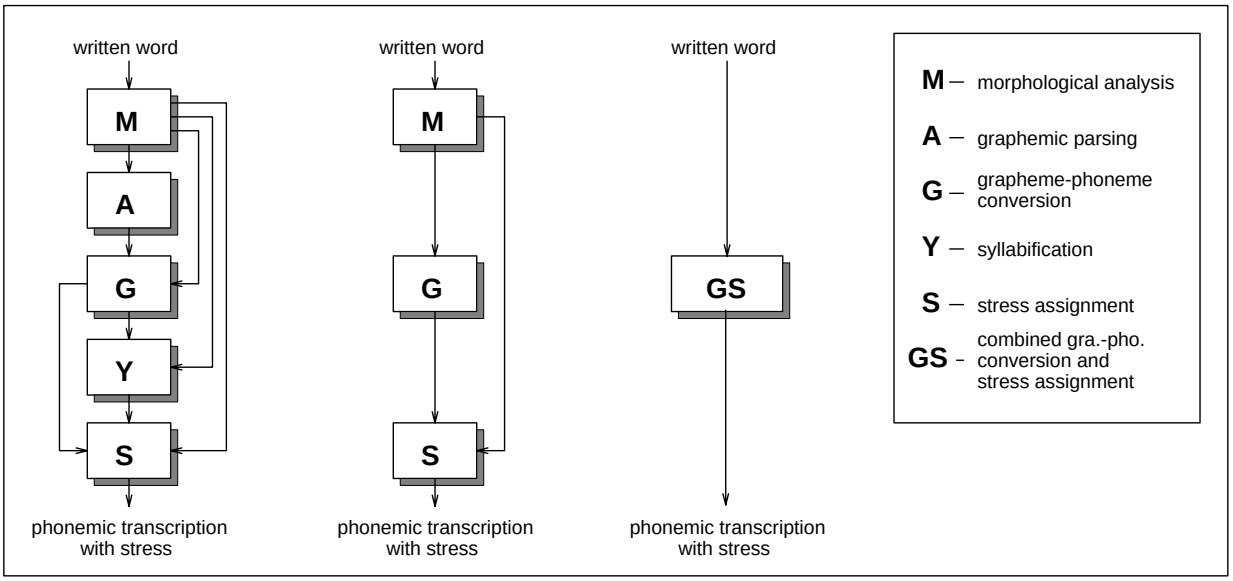


Figure 4: Architectures of the three investigated word-phonemisation systems. Left: M-A-G-Y-S; middle: M-G-S; right: GS. Rectangular boxes represent modules; the letter in the box corresponds to the subtask as listed in the legenda (far right). Arrows depict data flows from the raw input or a module, to a module or the output.

is possible on a moderate personal computer. The architectures of the two modular systems are displayed in Figure 4, which also displays the simple architecture of the system performing the GS task. We briefly outline both modular systems and report on the experiments needed for constructing them.

4.1.1 M-A-G-Y-S

The architecture of the M-A-G-Y-S system is inspired by SOUND1 (Hunnicutt, 1976; Hunnicutt, 1980), the word phonemisation subsystem of the MITALK text-to-speech system (Allen *et al.*, 1987). When the MITALK system is faced with an unknown word, SOUND1 produces on the basis of that word a phonemic transcription with stress markers (Allen *et al.*, 1987). This word phonemisation process is divided into the following five processing components:

1. *morphological segmentation*, which we implement as the module referred to as M;
2. *graphemic parsing*, module A;
3. *grapheme-phoneme conversion*, module G;
4. *syllabification*, module Y;

5. *stress assignment*, module S.

The architecture of the M-A-G-Y-S system is visualized in the left of Figure 4. It can be seen that the representations include direct output from previous modules, as well as representations from earlier modules. For example, the S module takes as input the syllable boundaries generated by the Y module, but also the phoneme string generated by the G module, and the morpheme boundaries generated by the M module.

M-A-G-Y-S is put to the test by applying IGTREE in 10-fold CV experiments to the five subtasks, connecting the modules after training, and measuring the combined score on correctly classified phonemes and stress markers, which is the desired output of the word-phonemisation system. An individual module can be trained on data from CELEX directly as input, but this method ignores the fact that modules in a working modular system can be expected to generate some amount of error. When one module generates an error, the subsequent module receives this error as input, assumes it is correct, and may generate another error. In a five-module system, this type of cascading errors may seriously hamper generalization accuracy. To counteract this potential disadvantage, modules can also be trained on the output of previous modules. Modules cannot be expected to learn to repair completely random, irregular errors, but whenever a previous module makes consistent errors on a specific input, this may be recognized by the subsequent module. Having detected a consistent error, the subsequent module is then able to repair the error and continue with successful processing. Earlier experiments performed on the tasks investigated in this paper have shown that classification errors on test instances are indeed consistently and significantly decreased when modules are trained on the output of previous modules rather than on data extracted directly from CELEX (Van den Bosch, 1997). Therefore, we train the M-A-G-Y-S system with IGTREE in the variant in which modules are trained on the output of other modules. We henceforth refer to this type of training as **adaptive** training, referring to the adaptation of a module to the errors of a previous module.

Figure 5 displays the results obtained with IGTREE under the adaptive variant of M-A-G-Y-S. The figure shows all percentages (displayed above the bars; error bars on top of the main bars indicate standard deviations) of incorrectly classified instances for each of the five subtasks, and a joint error on incorrectly classified phonemes with stress markers, which is the desired output of the system. The latter classification error, labelled PS in Figure 5, regards classification of an instance as incorrect if either or both of the phoneme and stress marker is incorrect. The figure shows that the joint error on phonemes and stress markers is 10.59% of test instances, on average. Computed in terms of transcribed words, only 35.89% of all test words are converted to stressed phonemic transcriptions flawlessly. The joint error is lower than the sum of the errors on the G subtask and the S subtask, 12.95%, suggesting that about 20% of the incorrectly classified test instances involve an incorrect classification of both the phoneme and the stress marker.

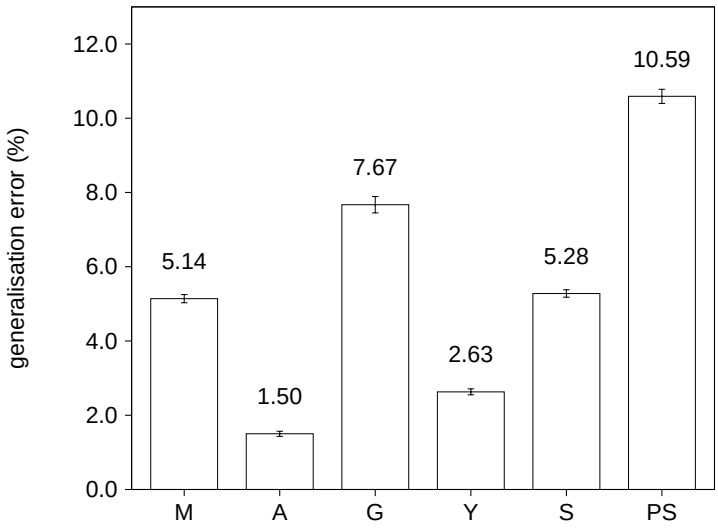


Figure 5: Generalization errors on the M-A-G-Y-S system in terms of the percentage of incorrectly classified test instances by IGTREE on the five subtasks M, A, G, Y, and S, and on phonemes and stress markers jointly (PS).

4.2 M-G-S

The subtasks of graphemic parsing (A) and grapheme-phoneme conversion (G) are clearly related. While A attempts to parse a letter string into graphemes, G converts graphemes to phonemes. Although they are performed independently in M-A-G-Y-S, they can be integrated easily when the class-'1'-instances of the A task are mapped to their associated phoneme rather than '1', and the class-'0'-instances are mapped to a phonemic null, /-/, rather than '0' (cf. Table 5). This task integration is also used in the NETtalk model (Sejnowski and Rosenberg, 1987). A similar argument can be made for integrating the syllabification and stress assignment modules into a single stress-assignment module. Stress markers, in our definition of the stress-assignment subtask, are placed solely on the positions which are also marked as syllable boundaries (i.e., on syllable-initial phonemes). Removing the syllabification subtask makes finding those syllable boundaries which are relevant for stress assignment an integrated part of stress assignment. Syllabification (Y) and stress assignment (S) can thus be integrated in a single stress-assignment module S.

When both pairs of modules are reduced to single modules, the three-module system M-G-S is obtained. Figure 4 displays the architecture of the M-G-S system in the middle. Experiments on this system are performed analogous to the experiments with the M-A-G-Y-S system; Figure 6 displays the average percentages of generalization errors generated by IGTREE on the three subtasks and phonemes and stress markers jointly (the error bar labelled PS).

Removing graphemic parsing (A) and syllabification (Y) as explicit in-between modules yields

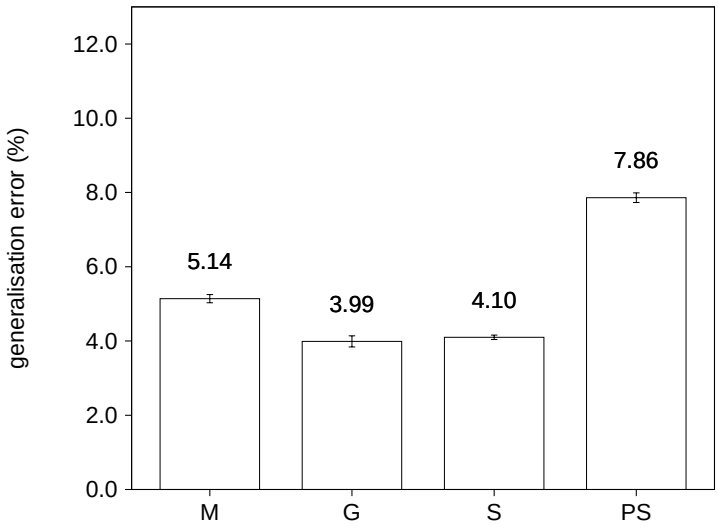


Figure 6: Generalization errors on the M-G-S system in terms of the percentage of incorrectly classified test instances by IGTREE on the three subtasks M, G, and S, and on phonemes and stress markers jointly (PS).

better accuracies on the grapheme-phoneme conversion (G) and stress assignment (S) subtasks than in the M-A-G-Y-S system. Both differences are significant; for G, ($t(19) = 43.70, p < 0.001$), and for S ($t(19) = 32.00, p < 0.001$). The joint accuracy on phonemes and stress markers is also significantly better in the M-G-S system than in the M-A-G-Y-S system ($t(19) = 37.50, p < 0.001$). Different from M-A-G-Y-S, the sum of the errors on phonemes and stress markers, 8.09%, is hardly more than the joint error on PSs, 7.86%: there is hardly an overlap in instances with incorrectly classified phonemes and with incorrectly placed stress markers. The percentage of flawlessly processed test words is 44.89%, which is markedly better than the 35.89% of M-A-G-Y-S.

4.2.1 Comparing M-A-G-Y-S, M-G-S, and GS

Figure 7 repeats the results obtained from applying IGTREE to each of the three systems, in terms of the percentages of incorrectly classified PSs. IGTREE yields significantly better generalization accuracy on phonemes and stress markers, both jointly and independently, trained on GS, as compared to M-A-G-Y-S and M-G-S. In terms of PSs, the accuracy on GS is significantly better than that of M-G-S with ($t(19) = 40.48, p < 0.001$), and that of M-A-G-Y-S with ($t(19) = 6.90, p < 0.001$). Thus, under our experimental conditions and using IGTREE as learning algorithm, optimal generalization accuracy on word phonemisation is obtained with GS, the system that does not incorporate any explicit decomposition of the word-phonemisation task.

As an additional comparison between the three systems, we analysed the positive and negative effects of learning the subtasks in their specific systems' context. The particular sequence of the five modules as in the M-A-G-Y-S system reflects a number of assumptions on the **utility** of

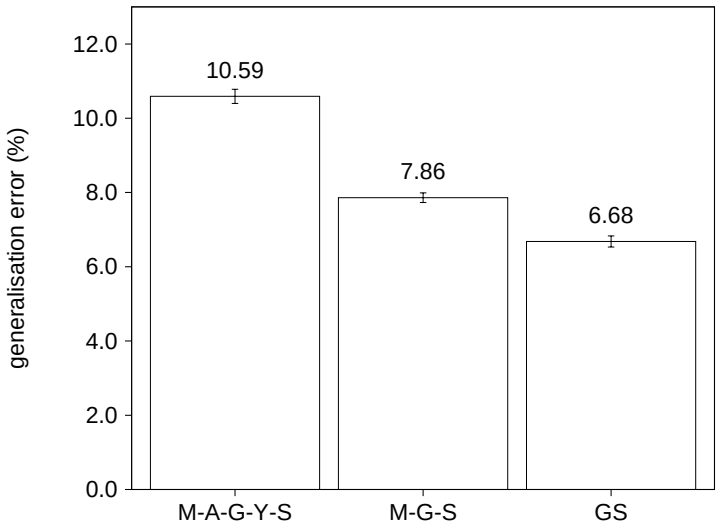


Figure 7: Generalization performances of IGTREE in terms of the percentage of incorrectly processed PSs, applied under the adaptive variant to the two modular systems M-A-G-Y-S and M-G-S, and applied to the single-module system GS.

using output from one subtask as input to another subtask. Morphological knowledge is useful as input to grapheme-phoneme conversion (e.g., to avoid pronouncing ph in loophole as /f/, or red in barred as /rɛd/); graphemic parsing is useful as input to grapheme-phoneme conversion (e.g., to avoid the pronunciation of gh in through); etc. Thus, feeding the output of a module A into a subsequent module B implies that one expects to perform better on module B with A's input than without. The accuracy results obtained with the modules of the M-A-G-Y-S, M-G-S, and GS systems can serve as tests for their respective underlying utility assumptions, when they are compared to the accuracies obtained with their subtasks learned in isolation.

To measure the utility effects of including the outputs of modules as inputs to other modules, we performed the following experiments:

1. We applied IGTREE in 10-fold CV experiments to each of the five subtasks M, A, G, Y, and S, only using letters (with the M, A, G, and S subtasks) or phonemes (with the Y and the S subtasks) as input, and their respective classification as output (cf. Table 5). The input is directly extracted from CELEX. These experiments provide the baseline score for each subtask, and are referred to as the **isolated** experiments.

2. We applied IGTREE in 10-fold CV experiments to all subtasks of the M-A-G-Y-S, M-G-S, and GS systems, training and testing on input extracted directly from CELEX. The results from these experiments reflect what would be the accuracy of the modular systems when each module would perform perfectly flawless. We refer to these experiments as **ideal**.

Table 6: Overview of utility effects of learning subtasks (M, A, G, Y, and S) as modules or partial tasks in the M-A-G-Y-S, M-G-S, and GS systems. For each module, in each system, the utility of training the module with ideal data (middle) and actual, modular data under the adaptive variant (right), is compared against the accuracy obtained with learning the subtasks in isolation (left). Accuracies are given in percentage of incorrectly classified test instances.

sub-	*% generalization error*				
task	*isolated*	*ideal*	*(utility)*	*actual*	*(utility)*
			M-A-G-Y-S		
M	5.14	5.14	(0.00)	5.14	(0.00)
A	1.39	1.66	(−0.27)	1.50	(−0.11)
G	3.72	3.68	(+0.04)	7.67	(−3.95)
Y	0.45	0.75	(−0.30)	2.63	(−2.16)
S	7.96	2.67	(+5.29)	5.28	(+2.68)
			M-G-S		
M	5.14	5.14	(0.00)	5.14	(0.00)
G	3.72	3.66	(+0.06)	3.99	(−0.27)
S	7.96	3.97	(+3.99)	4.10	(+3.86)
			GS		
G	3.72	–	–	3.79	(−0.07)
S	4.71	–	–	3.76	(+0.95)

With the results of these experiments we measure, for each subtask in each of the three systems, the utility effect of including the input of preceding modules, for the ideal case (with input straight from CELEX) as well as for the actual case (with input from preceding modules). A utility effect is the difference between IGTREE's generalization error on the subtask in modular context (either ideal or actual) and its accuracy on the same subtask in isolation. Table 6 lists all computed utility effects.

For the case of the M-A-G-Y-S system, it can be seen that the only large utility effect, even in the ideal case, could be obtained with the stress-assignment subtask. In the isolated case, the input consists of phonemes; in the M-A-G-Y-S system, the input contains morpheme boundaries, phonemes, and syllable boundaries. The ideal positive effect on the S module of 5.29% less errors turns out to be a positive effect of 2.68% in the actual system. The latter positive effect is outweighed by a rather large negative utility effect on the grapheme-phoneme conversion task of -3.95%. Both the A and Y subtasks do not profit from morphological boundaries as input, even in the ideal case; in the actual M-A-G-Y-S system, the utility effect of including morphological boundaries from M and phonemes from G in the syllabification module Y is markedly negative: -2.16%.

In the M-G-S system, the utility effects are generally less negative than in the M-A-G-Y-S system.

There is a small utility effect in the ideal case with including morphological boundaries as input to grapheme-phoneme conversion; in the actual M-G-S system, the utility effect is negative (-0.27%). The stress-assignment module benefits from including morphological boundaries and phonemes in its input, both in the ideal case and in the actual M-G-S system.

The GS system does not contain separate modules, but it is possible to compare the errors made on phonemes and stress assignments separately to the results obtained on the subtasks learned in isolation. Grapheme-phoneme conversion is learned with almost the same accuracy when learned in isolation as when learned as partial task of the GS task. Learning the grapheme-phoneme task, IGTREE is neither helped nor hampered significantly by learning stress assignment simultaneously. There is a positive utility effect in learning stress assignment, however. When stress assignment is learned in isolation with letters as input, IGTREE classifies 4.71% of test instances incorrectly, on average. (This is a lower error than obtained with learning stress assignment on the basis of phonemes, indicating that stress assignment should take letters as input rather than phonemes.) When the stress-assignment task is learned along with grapheme-phoneme conversion in the GS system, a marked improvement is obtained: 0.95% less classification errors are made.

Summarising, comparing the accuracies on modular subtasks to the accuracies on their isolated counterpart tasks shows only a few positive utility effects in the actual system, all obtained with stress assignment. The largest utility effect is found on the stress-assignment subtask of M-G-S. However, this positive utility effect does not lead to optimal accuracy on the S subtask; in the GS system, stress assignment is performed with letters as input, yielding the best accuracy on stress assignment in our investigations, viz. 3.76% incorrectly classified test instances.

5 Conclusion

We have presented a memory-based approach to learning word phonemisation. The approach is data-oriented and language-independent. Its only prerequisite is the presence of a word-pronunciation corpus, which are available for many languages. Furthermore, we have argued that a memory-based approach to learning word phonemisation should be favoured over other generic machine-learning approaches aimed at abstraction of learning material: we have argued and demonstrated that full memory of all training instances consistently yields optimal generalization accuracy. The more a learning algorithm attempts to abstract from individual learning instances, the more its generalization accuracy is harmed. We demonstrated this on a corpus of 77,565 English word-pronunciation pairs; IB1-IG, a memory-based learning algorithm with an information-gain weighted similarity metric, outperformed other algorithms and obtained a generalization accuracy of 93.5% correctly classified phonemes (including stress markers; 63.6% flawlessly converted test words).

A disadvantage of memory-based learning is its large computational memory demands and slow processing; we therefore developed IGTREE, a decision-tree learning algorithm which is an

approximate optimization of IB1-IG, IGTREE, which uses a fraction of IB1-IG's memory and is much faster; however, it performs slightly worse. Subsequently, we presented and tested TRIBL, a hybrid between IB1-IG and IGTREE, and demonstrated that this hybrid could indeed maintain IB1-IG's performance while processing at the speed of IGTREE. Memory-based learning can be optimized, albeit carefully, by finding an optimal hybrid of decision-tree learning of the most relevant features of the data, and memory-based learning of the remaining features.

Finally, we demonstrated that the explicit modelling of abstraction levels as sequentially-trained memory-based modules, as suggested by mainstream morpho-phonological theories, does not improve the performance on word phonemisation. The best performance is obtained by training a learning algorithm on the direct conversion of grapheme strings to a combined class representing phonemic mapping and stress assignment simultaneously. Although positive utility effects of sequenced modules are found, they are overshadowed by propagated errors through the modules; the more modules in the system, the more the overall output of the system suffers from these unwanted errors. Within the scope of our experimental settings, we can conclude that abstraction levels should best be left explicit. Our results strongly suggest that most relevant information for phonetic mapping and stress assignment is stored implicitly in the written words themselves, in relatively local areas.

Further research is needed to measure the effects of widening the window used for generating instances. The assumption that a window of 3-1-3 (three left-context graphemes, a focus grapheme, and three right-context graphemes) is sufficient and adequate, is rather dangerous, since a considerable number of word-pronunciation instances remain ambiguous with this context. Other areas of relevant future investigation include using transcribed text as training material, using previous classifications as input features, learning meta-word phenomena such as intonation contours and sentence accents, and incorporating trained GS-modules (e.g., optimized TRIBLs) in text-to-speech synthesis applications.

Acknowledgements

This research was done in the context of the "Induction of Linguistic Knowledge" research programme, partially supported by the Foundation for Language Speech and Logic (TSL), which is funded by the Netherlands Organization for Scientific Research (NWO). Part of the second author's work was performed at the Department of Computer Science of the Universiteit Maastricht. We thank Jakub Zavrel, Bertjan Busser, the other members of the Tilburg ILK group, Ton Weijters, and Eric Postma for fruitful discussions and feedback.

References

Aha D. W. (ed.) (1997). *Lazy learning.* Dordrecht: Kluwer Academic Publishers. reprinted from: *Artificial Intelligence Review*, **11**:1–5.

Aha D. W., Kibler D., and Albert M. (1991). Instance-based learning algorithms. *Machine Learning*, Vol. 7, pp. 37–66.

Allen J., Hunnicutt S., and Klatt D. (1987). *From text to speech: The* MITalk *system.* Cambidge University Press, Cambridge, UK.

Andersen O. and Dalsgaard P. (1994). A self-learning approach to transcription of Danish proper names. *Proceedings of the International Conference on Spoken Language Processes*, pp. 1627–1630.

Burnage G. (1990). CELEX*: A guide for users.* Centre for Lexical Information, Nijmegen.

Chomsky N. and Halle M. (1968). *The sound pattern of English.* Harper and Row, New York, NY.

Cost S. and Salzberg S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, Vol. 10, pp. 57–78.

Cover T. M. and Hart P. E. (1967). Nearest neighbor pattern classification. *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, Vol. 13, pp. 21–27.

Daelemans W. (1988). GRAFON: A grapheme-to-phoneme system for Dutch. *Proceedings Twelfth International Conference on Computational Linguistics (COLING-88), Budapest*, pp. 133–138.

Daelemans W. (1996). Experience-driven language acquisition and processing. *Proceedings of the CLS Opening Academic Year 1996-1997* (eds. M. Van der Avoird and C. Corsius), pp. 83–95. Tilburg: CLS.

Daelemans W. and Van den Bosch A. (1992). Generalisation performance of backpropagation learning on a syllabification task. *TWLT3: Connectionism and Natural Language Processing* (eds. M. F. J. Drossaers and A. Nijholt), pp. 27–37, Enschede. Twente University.

Daelemans W. and Van den Bosch A. (1993). TabTalk: reusability in data-oriented grapheme-to-phoneme conversion. *Proceedings of Eurospeech '93*, pp. 1459–1466, Berlin. T.U. Berlin.

Daelemans W. and Van den Bosch A. (1996). Language-independent data-oriented grapheme-to-phoneme conversion. *Progress in Speech Processing* (eds. J. P. H. Van Santen, R. W. Sproat, J. P. Olive, and J. Hirschberg), pp. 77–89. Berlin: Springer-Verlag.

Daelemans W., Van den Bosch A., and Weijters A. (1997a). IGTree: using trees for classification in lazy learning algorithms. *Artificial Intelligence Review*, Vol. 11, pp. 407–423.

Daelemans W., Van den Bosch A., and Zavrel J. (1997b). A feature-relevance heuristic for indexing and compressing large case bases. *Poster Papers of the Ninth European Conference on Machine Learing* (eds. M. Van Someren and G. Widmer), pp. 29–38, Prague, Czech Republic. University of Economics.

Damper R. I. (1995). Self-learning and connectionist approaches to text-phoneme conversion. *Connectionist Models of Memory and Language* (eds. J. Levy, D. Bairaktaris, J. Bullinaria, and P. Cairns), pp. 117–144. London, UK: UCL Press.

Damper R. I. and Eastmond J. F. G. (1997). Pronunciation by analogy: impact of implementational choices on performance. *Language and Speech*, Vol. 40, pp. 1–23.

Dasarathy B. V. (1980). Nosing around the neighborhood: A new system structure and classification rule for recognition in partially exposed environments. *Pattern Analysis and Machine Intelligence*, Vol. 2, pp. 67–71.

Dedina M. J. and Nusbaum H. C. (1991). PRONOUNCE: a program for pronunciation by analogy. *Computer Speech and Language*, Vol. 5, pp. 55–64.

Dietterich T. G., Hild H., and Bakiri G. (1995). A comparison of ID3 and Backpropagation for English text-to-speech mapping. *Machine Learning*, Vol. 19, No. 1, pp. 5–28.

Gates G. W. (1972). The reduced nearest neighbor rule. *IEEE Transactions on Information Theory*, Vol. 18, pp. 431–433.

Glushko R. J. (1979). The organisation and activation of orthographic knowledge in reading aloud. *Journal of Experimental Psychology: Human Perception and Performance*, Vol. 5, pp. 647–691.

Goldsmith J. (1976). An overview of autosegmental phonology. *Linguistic Analysis*, Vol. 2, pp. 23–68.

Hart P. E. (1968). The condensed nearest neighbor rule. *IEEE Transactions on Information Theory*, Vol. 14, pp. 515–516.

Hunnicutt S. (1976). Phonological rules for a text-to-speech system. *American Journal of Computational Linguistics*, Vol. Microfiche 57, pp. 1–72.

Hunnicutt S. (1980). Grapheme-phoneme rules: a review. Technical Report STL QPSR 2-3, Speech Transmission Laboratory, KTH, Sweden.

Kohonen T. (1986). Dynamically expanding context, with application to the correction of symbol strings in the recognition of continuous speech. *Proceedings of the Eighth International Conference on Pattern Recognition*, pp. 27–31, Paris, France.

Koskenniemi K. (1984). A general computational model for wordform recognition and production. *Proceedings of the Tenth International Conference on Computational Linguistics / 22nd Annual Conference of the ACL*, pp. 178–181.

Liberman M. and Prince A. (1977). On stress and linguistic rhythm. *Linguistic Inquiry*, , No. 8, pp. 249–336.

Ling C. X. and Wang H. (1996). A decision-tree model for reading aloud with automatic alignment and grapheme generation. submitted.

Lucassen J. M. and Mercer R. L. (1984). An information theoretic approach to the automatic determination of phonemic baseforms. *Proceedings of* ICASSP *'84, San Diego*, pp. 42.5.1–42.5.4.

Luk R. W. P. and Damper R. I. (1996). Stochastic phonographic transduction for English. *Computer Speech and Language*, Vol. 10, pp. 133–153.

Mohanan K. P. (1986). *The theory of lexical phonology*. Dordrecht: D. Reidel.

Pirelli V. and Federici S. (1994). On the pronunciation of unknown words by analogy in text-to-speech systems. *Proceedings of the Second Onomastica Research Colloquium*, London.

Quinlan J. R. (1986). Induction of decision trees. *Machine Learning*, Vol. 1, pp. 81–206.

Quinlan J. R. (1993). C4.5*: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann.

Rumelhart D. E., Hinton G. E., and Williams R. J. (1986). Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (eds. D. E. Rumelhart and J. L. McClelland), Vol. 1: Foundations, pp. 318–362. Cambridge, MA: The MIT Press.

Sejnowski T. J. and Rosenberg C. S. (1987). Parallel networks that learn to pronounce English text. *Complex Systems*, Vol. 1, pp. 145–168.

Stanfill C. (1987). Memory-based reasoning applied to English pronunciation. *Proceedings of the Sixth National Conference on Artificial Intelligence (*AAAI-87*)*, pp. 577–581. Los Altos, CA: Morgan Kaufmann.

Stanfill C. and Waltz D. (1986). Toward memory-based reasoning. *Communications of the* ACM, Vol. 29, No. 12, pp. 1213–1228.

Sullivan K. P. H. and Damper R. I. (1992). Novel-word pronunciation with a text-to-speech system. *Talking machines: theories, models, and applications* (eds. G. Bailly and C. Benoît), pp. 183–195. Amsterdam: Elsevier.

Sullivan K. P. H. and Damper R. I. (1993). Novel-word pronunciation: a cross-language study. *Speech Communication*, Vol. 13, pp. 441–452.

Torkkola K. (1993). An efficient way to learn English grapheme-to-phoneme rules automatically. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (*ICASSP*)*, Vol. 2, pp. 199–202, Minneapolis.

Van Coile B. (1990). Inductive learning of grapheme-to-phoneme rules. *Proceedings of the International Conference on Spoken Language Processes 1990*, Vol. 2, pp. 765–768, Kobe, Japan.

Van den Bosch A. and Daelemans W. (1993). Data-Oriented Methods for Grapheme-to-Phoneme Conversion. *Proceedings of the 6th Conference of the EACL*, pp. 45–53.

Van den Bosch A., Daelemans W., and Weijters A. (1996). Morphological analysis as classification: an inductive-learning approach. *Proceedings of NeMLaP-2, Ankara, Turkey* (eds. K. Oflazer and H. Somers), pp. 79–89.

Van den Bosch A. (1997). *Learning to pronounce written words, a study in inductive language learning.* PhD thesis, Universiteit Maastricht.

Van den Bosch A., Weijters A., Van den Herik H. J., and Daelemans W. (1997b). When small disjuncts abound, try lazy learning. To appear in the Proceedings of BENELEARN-97, Tilburg University.

Van der Wouden T. (1990). Celex: Building a multifunctional polytheoretical lexical data base. *Proceedings of BudaLex'88* (ed. T. Magay). Budapest: Akadémiai Kladó.

Viterbi A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE *Transactions on Information Theory*, Vol. 13, pp. 260–269.

Weijters A. (1991). A simple look-up procedure superior to NETtalk? *Proceedings of* ICANN*-91, Espoo, Finland.*

Wolpert D. H. (1990). Constructing a generalizer superior to NETtalk via a mathematical theory of generalization. *Neural Networks,* Vol. 3, pp. 445–452.

Yvon F. (1996). *Prononcer par analogie: motivation, formalisation et évaluation.* PhD thesis, Ecole Nationale Supérieure des Télécommunication, Paris.